Working with Inheritance - 0

What will be the result of compiling and executing Test class?

```
//Test.java
abstract class Animal {
     private String name;

     Animal(String name) {
        this.name = name;
     }

     public String getName() {
         return name;
     }
}

class Dog extends Animal {
     private String breed;
     Dog(String breed) {
         this.breed = breed;
     }

     Dog(String name, String breed) {
         super(name);
         this.breed = breed;
     }

     public String getBreed() {
         return breed;
     }
}

public class Test {
     public static void main(String[] args) {
         Dog dog1 = new Dog("Beagle");
         Dog dog2 = new Dog("Bubbly", "Poodle");
         System.out.println(dog1.getName() + ":" + dog1.getBreed() +
        ":" +
                            dog2.getName() + ":" + dog2.getBreed());
     }
}
```

A - Compilation error for Animal Class

B - Compilation error for Animal(String) constructor

C - Compilation error for Dog(String) constructor

D - Compilation error for Dog(String, String) constructor

E - null:Beagle:Bubbly:Poodle

F - :Beagle:Bubbly:Poodle

Working with Inheritance - 1

Consider code below:

```java
class PenDrive {
    int capacity;
    PenDrive(int capacity) {
        this.capacity = capacity;
    }
}

class OTG extends PenDrive {
    String type;
    String make;
    OTG(int capacity, String type) {
        /*INSERT-1*/
    }
    OTG(String make) {
        /*INSERT-2*/
        this.make = make;
    }
}

public class Test {
    public static void main(String[] args) {
        OTG obj = new OTG(128, "TYPE-C");
        System.out.println(obj.capacity + ":" + obj.type);
    }
}
```

Currently above code causes compilation error.

Which of the options can successfully print 128:TYPE-C on to the console?

A - None of the other options

B -

```java
// Replace /*INSERT-1*/ with:
super.capacity = capacity;
this.type = type;
// Replace /*INSERT-2*/ with:
super(128);
```

C -

```java
// Replace /*INSERT-1*/ with:
super(capacity);
// Replace /*INSERT-2*/ with:
super(128);
```

D -

```java
// Replace /*INSERT-1*/ with:
```

```java
super(capacity);
this.type = type;
// Replace /*INSERT-2*/ with:
super(0);
```

E-

```java
// Replace /*INSERT-1*/ with:
this.type = type;
super(capacity);
// Replace /*INSERT-2*/ with:
super(128);
```

Working with Inheritance - 2

Consider codes below:

```
//A.java
package com.training.oca;

public class A {
     public int i1 = 1;
     protected int i2 = 2;
}


//B.java
package com.training.oca.test;

import com.training.oca.A;

public class B extends A {
     public void print() {
         A obj = new A();
         System.out.println(obj.i1); //Line 8
         System.out.println(obj.i2); //Line 9
         System.out.println(this.i2); //Line 10
         System.out.println(super.i2); //Line 11
     }

     public static void main(String [] args) {
         new B().print();
     }
}
```

One of the statements inside print() method is causing compilation failure. Which of the below solutions will help to resolve compilation error?

A - Comment the statement at Line 9

B - Comment the statement at Line 10

C - Comment the statement at Line 11

D - Comment the statement at Line 8

Working with Inheritance - 3

Consider codes below:

```
//A.java
package com.training.oca;

public class A {
     public void print() {
         System.out.println("A");
     }
}


//B.java
package com.training.oca;

public class B extends A {
     public void print() {
         System.out.println("B");
     }
}


//Test.java
package com.training.oca.test;

import com.training.oca.*;

public class Test {
     public static void main(String[] args) {
         A obj1 = new A();
         B obj2 = (B)obj1;
         obj2.print();
     }
}
```

What will be the result of compiling and executing Test class?

A - ClassCastException is thrown at runtime

B - Compilation error

C - A

D - B

Working with Inheritance - 4

Consider below code fragment:

```
interface Printable {
    public void setMargin();
    public void setOrientation();
}

abstract class Paper implements Printable { //Line 7
    public void setMargin() {}
    //Line 9
}

class NewsPaper extends Paper { //Line 12
    public void setMargin() {}
    //Line 14
}
```

Above code is currently giving compilation error. Which 2 modifications, done independently, enable the code to compile?

A - Replace the code at Line 7 with: class Paper implements Printable {

B - Insert at Line 9: public abstract void setOrientation();

C - Replace the code at Line 12 with: abstract class NewsPaper extends Paper {

D - Insert at Line 14: public void setOrientation() {}

Working with Inheritance - 5

Which of these access modifiers can be used for a top level interface?

A - All of the other options

B - private

C - public

D - protected

Working with Inheritance - 6

What will be the result of compiling and executing Test class?

```java
//Test.java
class Parent {
     int i = 10;
     Parent(int i) {
         super();
         this.i = i;
     }
}

class Child extends Parent {
     int j = 20;

     Child(int j) {
         super(0);
         this.j = j;
     }

     Child(int i, int j) {
         super(i);
         this(j);
     }

}

public class Test {
     public static void main(String[] args) {
         Child child = new Child(1000, 2000);
         System.out.println(child.i + ":" + child.j);
     }
}
```

A - Compilation error for Test class

B - 1000:2000

C - Compilation error for Child(int) constructor

D - Compilation error for Child(int, int) Constructor

E - 1000:0

F - Compilation error for Parent(int) constructor

Working with Inheritance - 7

Which of these keywords can be used to prevent inheritance of a class?

A - constant

B - class

C - final

D - super

Working with Inheritance - 8

Consider below code:

```java
//Test.java
class Parent {
    public String toString() {
        return "Inner ";
    }
}

class Child extends Parent {
    public String toString() {
        return super.toString().concat("Peace!");
    }
}

public class Test {
    public static void main(String[] args) {
        System.out.println(new Child());
    }
}
```

What will be the result of compiling and executing Test class?

A - Peace!

B - Compilation error

C - Inner Peace!

D - Inner

Working with Inheritance - 9

Given code of LogHelper.java file:

```java
abstract class Helper {
    int num = 100;
    String operation = null;

    protected abstract void help();

    void log() {
        System.out.println("Helper-log");
    }
}

public class LogHelper extends Helper {
    private int num = 200;
    protected String operation = "LOGGING";

    void help() {
        System.out.println("LogHelper-help");
    }

    void log() {
        System.out.println("LogHelper-log");
    }

    public static void main(String [] args) {
        new LogHelper().help();
    }
}
```

Which of the following changes, done independently, allows the code to compile and on execution prints LogHelper-help?

Select ALL that apply.

A - Remove the private modifier from the num variable of LogHelper class

B - Add the public modifier to the help() method of LogHelper class

C - Remove the protected modifier from the help() method of Helper class

D - Add the protected modifier to the log() method of Helper class

E - Add the protected modifier to the help() method of LogHelper class

F - Add the protected modifier to the log() method of LogHelper class

G - Add the public modifier to the log() method of LogHelper class

H - Remove the protected modifier from the operation variable of LogHelper class

Working with Inheritance - 10

What will be the result of compiling and executing Test class?

```
class Vehicle {
    public int getRegistrationNumber() {
        return 1;
    }
}

class Car {
    public int getRegistrationNumber() {
        return 2;
    }
}

public class Test {
    public static void main(String[] args) {
        Vehicle obj = new Car();
        System.out.println(obj.getRegistrationNumber());
    }
}
```

A - Compilation error

B - 2

C - 1

D - An exception is thrown at runtime

Working with Inheritance - 11

Which is not a valid statement based on given code?

```java
class A{}
class B extends A{}
```

A - A a = new B();

B - B a = new B();

C - A a = new A();

D - B b = new A();

Working with Inheritance - 12

What will be the result of compiling and executing Test class?

```
class M { }
class N extends M { }
class O extends N { }
class P extends O { }

public class Test {
     public static void main(String args []) {
         M obj = new O();
         if(obj instanceof M)
           System.out.print("M");
         if(obj instanceof N)
           System.out.print("N");
         if(obj instanceof O)
           System.out.print("O");
         if(obj instanceof P)
           System.out.print("P");
     }
}
```

A - MNP

B - MNO

C - MOP

D - NOP

Working with Inheritance - 13

For the given code:

```java
interface I01 {
    void m1();
}

public class Implementer extends Object implements I01{
    protected void m1() {

    }
}
```

A - interface I01 gives compilation error as method m1 is not public.

B - None of the other options.

C - Implementer class declaration is not correct.

D - Method m1() in Implementer class is not implemented correctly.

Working with Inheritance - 14

What will be the result of compiling and executing Test class?

```
class Super {
    public Super(int i) {
        System.out.println(100);
    }
}

class Sub extends Super {
    public Sub() {
        System.out.println(200);
    }
}

public class Test {
    public static void main(String[] args) {
        new Sub();
    }
}
```

A - 200

B - Compilation Error

C - 100 200

D - 200 100

Working with Inheritance - 15

Which one of these top level classes cannot be sub-classed?

A - final class Electronics {}

B - class Dog {}

C - private class Car {}

D - abstract class Cat {}

Working with Inheritance - 16

super keyword in java is used to:

A - refer to parent class object.

B - refer to static method of the class.

C - refer to current class object.

D - refer to static variable of the class.

Working with Inheritance - 17

What will be the result of compiling and executing TestBaseDerived class?

```java
//TestBaseDerived.java
class Base {
     protected void m1() {
         System.out.println("Base: m1()");
     }
}

class Derived extends Base {
     void m1() {
         System.out.println("Derived: m1()");
     }
}

public class TestBaseDerived {
     public static void main(String[] args) {
         Base b = new Derived();
         b.m1();
     }
}
```

A - Derived: m1()

B - Base: m1()

C - Base: m1() Derived: m1()

D - None of the other options

Working with Inheritance - 18

Consider codes below:

```
//A.java
package com.training.oca;

public class A {
     public void print() {
         System.out.println("A");
     }
}


//B.java
package com.training.oca;

public class B extends A {
     public void print() {
         System.out.println("B");
     }
}


//C.java
package com.training.oca;

public class C extends A {
     public void print() {
         System.out.println("C");
     }
}


//Test.java
package com.training.oca.test;

import com.training.oca.*;

public class Test {
     public static void main(String[] args) {
         A obj1 = new C();
         A obj2 = new B();
         C obj3 = (C)obj1;
         C obj4 = (C)obj2;
         obj3.print();
     }
}
```

What will be the result of compiling and executing Test class?

A - B

B - Compilation error

C - ClassCastException is thrown at runtime

D - A

E - C

Working with Inheritance - 19

Given the following definitions of the class Insect and the interface Flyable, the task is to declare a class Mosquito that inherits from the class Insect and implements the interface Flyable.

```
class Insect {}
interface Flyable {}
```

Select the correct option to accomplish this task:

A -

```
class Mosquito implements Insect extends Flyable{}
```

B -

```
class Mosquito implements Insect, Flyable{}
```

C -

```
class Mosquito extends Insect, Flyable{}
```

D -

```
class Mosquito extends Insect implements Flyable{}
```

Working with Inheritance - 20

Consider below code of Test.java file:

```java
class Document {
    int pages;
    Document(int pages) {
        this.pages = pages;
    }
}

class Word extends Document {
    String type;
    Word(String type) {
        super(20); //default pages
        /*INSERT-1*/
    }

    Word(int pages, String type) {
        /*INSERT-2*/
        super.pages = pages;
    }
}

public class Test {
    public static void main(String[] args) {
        Word obj = new Word(25, "TEXT");
        System.out.println(obj.type + "," + obj.pages);
    }
}
```

Currently above code causes compilation error.

Which of the options can successfully print TEXT,25 on to the console?

A -

```java
// Replace /*INSERT-1*/ with:
super.type = type;
// Replace /*INSERT-2*/ with:
super(type);
```

B -

```java
// Replace /*INSERT-1*/ with:
this.type = type;
// Replace /*INSERT-2*/ with:
this(type);
```

C - None of the other options

D -

```java
// Replace /*INSERT-1*/ with:
```

```java
super.type = type;
// Replace /*INSERT-2*/ with:
this(type);
```

E -

```java
// Replace /*INSERT-1*/ with:
this(type);
// Replace /*INSERT-2*/ with:
this.type = type;
```

Working with Inheritance - 21

What will be the result of compiling and executing Circus

```java
//Circus.java
class Animal {
    protected void jump() {
        System.out.println("Animal");
    }
}

class Cat extends Animal {
    public void jump(int a) {
        System.out.println("Cat");
    }
}

class Deer extends Animal {
    public void jump() {
        System.out.println("Deer");
    }
}

public class Circus {
    public static void main(String[] args) {
        Animal cat = new Cat();
        Animal deer = new Deer();
        cat.jump();
        deer.jump();
    }
}
```

A - Animal Deer

B - Animal Animal

C - Cat Deer

D - Cat Animal

Working with Inheritance - 22

What will be the result of compiling and executing Test class?

```
class A {
     A() {
         this(1);
         System.out.println("M");
     }

     A(int i) {
         System.out.println("N");
     }
}

class B extends A {

}

public class Test {
    public static void main(String[] args) {
         new B();
     }
}
```

A - N M

B - M

C - N

D - M N

Working with Inheritance - 23

Given code of Test.java file:

```java
class Base {
    static void print() { //Line n1
        System.out.println("BASE");
    }
}

class Derived extends Base {
    static void print() { //Line n2
        System.out.println("DERIVED");
    }
}

public class Test {
    public static void main(String[] args) {
        Base b = null;
        Derived d = (Derived) b; //Line n3
        d.print(); //Line n4
    }
}
```

Which of the following statements is true for above code?

A - Line n2 causes compilation error

B - Code compiles successfully and on execution Line n3 throws an exception

C - Line n4 causes compilation error

D - Code compiles successfully and on execution prints BASE on to the console

E - Code compiles successfully and on execution prints DERIVED on to the console

F - Line n3 causes compilation error

Working with Inheritance - 24

Consider below code of Test.java file:

```java
class Shape {
    int side = 0; //Line n1

    int getSide() { //Line n2
        return side;
    }
}

class Square extends Shape {
    private int side = 4; //Line n3

    protected int getSide() { //Line n4
        return side;
    }
}

public class Test {
    public static void main(String[] args) {
        Shape s = new Square();
        System.out.println(s.side + ":" + s.getSide());
    }
}
```

What will be the result of compiling and executing above code?

A - 0:4

B - Compilation error at Line n4

C - Compilation error at Line n3

D - 4:0

E - 4:4

F - 0:0

Working with Inheritance - 25

Consider below codes of 3 java files:

```
//Animal.java
package a;

public class Animal {
    Animal() {
        System.out.print("ANIMAL-");
    }
}


//Dog.java
package d;

import a.Animal;

public class Dog extends Animal {
    public Dog() {
        System.out.print("DOG");
    }
}


//Test.java
package com.training.oca;

import d.Dog;

public class Test {
    public static void main(String[] args) {
        new Dog();
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error in Test.java file

B - It executes successfully and prints ANIMAL-DOG on to the console

C - Compilation error in Animal.java file

D - It executes successfully but nothing is printed on to the console

E - It executes successfully and prints DOG on to the console

F - Compilation error in Dog.java file

Working with Inheritance - 26

Consider below code of Test.java file:

```
class Super {
    void Super() {
        System.out.print("KEEP_");
    }
}

class Base extends Super {
    Base() {
        Super();
        System.out.print("GOING_");
    }
}

public class Test {
    public static void main(String[] args) {
        new Base();
    }
}
```

What will be the result of compiling and executing above code?

A - Compilation Error in Base class

B - Compilation Error in Super class

C - It prints KEEP_KEEP_GOING_ on to the console

D - Compilation Error in Test class

E - It prints GOING_KEEP_ on to the console

F - It prints GOING_ on to the console

G - It prints KEEP_GOING_ on to the console

Working with Inheritance - 27

Given code of Test.java file:

```
class X {
    void greet() {
        System.out.println("Good Morning!");
    }
}

class Y extends X {
    void greet() {
        System.out.println("Good Afternoon!");
    }
}

class Z extends Y {
    void greet() {
        System.out.println("Good Night!");
    }
}

public class Test {
    public static void main(String[] args) {
        X x = new Z();
        x.greet(); //Line n1
        ((Y)x).greet(); //Line n2
        ((Z)x).greet(); //Line n3
    }
}
```

What will be the result of compiling and executing above code?

A - An exception is thrown at runtime

B - Compilation error

C - It compiles successfully and on execution prints below:

```
Good Morning!
Good Morning!
Good Morning!
```

D - It compiles successfully and on execution prints below:

```
Good Night!
Good Afternoon!
Good Morning!
```

E - It compiles successfully and on execution prints below:

```
Good Night!
Good Night!
Good Night!
```

Working with Inheritance - 28

Consider below codes of 3 java files:

```
//M.java
package com.training.oca;

public class M {
    public void printName() {
        System.out.println("M");
    }
}

//N.java
package com.training.oca;

public class N extends M {
    public void printName() {
        System.out.println("N");
    }
}

//Test.java
package com.training.oca.test;

import com.training.oca.*;

public class Test {
    public static void main(String[] args) {
        M obj1 = new M();
        N obj2 = (N)obj1;
        obj2.printName();
    }
}
```

What will be the result of compiling and executing Test class?

A - An exception is thrown at runtime

B - Compilation error

C - It executes successfully and prints N on to the console

D - It executes successfully and prints M on to the console

Working with Inheritance - 29

Consider below code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        P p = new R(); //Line n1
        System.out.println(p.compute("Go")); //Line n2
    }
}

class P {
    String compute(String str) {
        return str + str + str;
    }
}

class Q extends P {
    String compute(String str) {
        return super.compute(str.toLowerCase());
    }
}

class R extends Q {
    String compute(String str) {
        return super.compute(str.replace('o', 'O')); //2nd argument
        is uppercase O
    }
}
```

What will be the result of compiling and executing Test class?

A - GO

B - gOgOgoO

C - GOGOGO

D - Go

E - go

F - GoGoGo

G - gogogo

Working with Inheritance - 30

Given code of Test.java file:

```java
class M {
    public void main(String[] args) { //Line n1
        System.out.println("M");
    }
}

class N extends M {
    public static void main(String[] args) { //Line n2
        new M().main(args); //Line n3
    }
}

public class Test {
    public static void main(String[] args) {
        N.main(args); //Line n4
    }
}
```

Which of the following statements is true for above code?

A - Line n2 causes compilation error

B - Line n4 causes compilation error

C - Line n1 causes compilation error

D - Line n3 causes compilation error

E - It executes successfully and prints M on to the console

Working with Inheritance - 31

Given code of Test.java file:

```java
class Parent {
    int var = 1000; // Line n1

    int getVar() {
        return var;
    }
}

class Child extends Parent {
    private int var = 2000; // Line n2

    int getVar() {
        return super.var; //Line n3
    }
}

public class Test {
    public static void main(String[] args) {
        Child obj = new Child(); // Line n4
        System.out.println(obj.var); // Line n5
    }
}
```

There is a compilation error in the code.

Which three modifications, done independently, print 1000 on to the console?

A - Change Line n1 to private int var = 1000;

B - Delete the Line n2

C - Change Line n4 to Parent obj = new Child();

D - Change Line n3 to return var;

E - Delete the method getVar() from the Child class

F - Change Line n5 to System.out.printin(obj.getVar());

Working with Inheritance - 32

Consider below code fragment:

```
abstract class Food {
    protected abstract double getCalories();
}

class JunkFood extends Food {
    double getCalories() {
        return 200.0;
    }
}
```

Which 3 modifications, done independently, enable the code to compile?

A - Remove the protected access modifier from the getCalories() method of Food class

B - Make the getCalories() method of JunkFood class private

C - Make the getCalories() method of Food class public

D - Make the getCalories() method of Food class private

E - Make the getCalories() method of JunkFood class protected

F - Make the getCalories() method of JunkFood class public

Working with Inheritance - 33

Given code of Test.java file:

```java
interface X1 {
    default void print() {
        System.out.println("X1");
    }
}

interface X2 extends X1 {
    void print();
}

interface X3 extends X2 {
    default void print() {
        System.out.println("X3");
    }
}

class X implements X3 {}

public class Test {
    public static void main(String[] args) {
        X1 obj = new X();
        obj.print();
    }
}
```

Which of the following statements is correct?

A - interface X2 fails to compile

B - class Test fails to compile

C - class Test compiles successfully and on execution prints X1 on to the console

D - class Test compiles successfully and on execution prints X3 on to the console

E - interface X1 fails to compile

F - class X fails to compile

G - interface X3 fails to compile

Working with Inheritance - 34

Consider below codes of 4 java files:

```java
//Moveable.java
package com.training.oca;

public interface Moveable {
    void move();
}


//Animal.java
package com.training.oca;

public abstract class Animal {
    void move() {
        System.out.println("ANIMAL MOVING");
    }
}


//Dog.java
package com.training.oca;

public class Dog extends Animal implements Moveable {}


//Test.java
package com.training.oca;

public class Test {
    public static void main(String[] args) {
        Moveable moveable = new Dog();
        moveable.move();
    }
}
```

Which of the following statements is correct?

A - There is a compilation error in Dog.java file

B - There is no compilation error and on execution, Test class prints ANIMAL MOVING on to the console

C - There is a compilation error in Animal.java file

D - There is a compilation error in Test.java file

Working with Inheritance - 35

Consider below codes of 3 java files:

```
//Sellable.java
package com.training.oca;

public interface Sellable {
    double getPrice();

    default String symbol() {
        return "$";
    }
}

//Chair.java
package com.training.oca;

public class Chair implements Sellable {
    public double getPrice() {
        return 35;
    }
    public String symbol() {
        return "£";
    }
}

//Test.java
package com.training.oca;

public class Test {
    public static void main(String[] args) {
        Sellable obj = new Chair(); //Line n1
        System.out.println(obj.symbol() + obj.getPrice()); //Line n2
    }
}
```

What will be the result of compiling and executing Test class?

A - It compiles successfully and on execution prints $35.00 on to the console

B - Compilation error in Chair class

C - It compiles successfully and on execution prints $35.0 on to the console

D - It compiles successfully and on execution prints £35 on to the console

E - It compiles successfully and on execution prints $35 on to the console

F - It compiles successfully and on execution prints £35.00 on to the console

G - It compiles successfully and on execution prints £35.0 on to the console

H - Compilation error in Test class

Working with Inheritance - 36

Consider below codes of 2 java files:

```java
//Counter.java
package com.training.oca;

public interface Counter {
    int count = 10; //Line n1
}

//Test.java
package com.training.oca;

public class Test {
    public static void main(String[] args) {
        Counter [] arr = new Counter[2]; //Line n2
        for(Counter ctr : arr) {
            System.out.print(ctr.count); //Line n3
        }
    }
}
```

Which of the following statements is correct?

A - Test class compiles successfully and on execution prints 1010 on to the console

B - Line n3 throws an exception at runtime

C - Only Line n2 causes compilation error

D - Line n1 and Line n2 cause compilation error

E - Only Line n3 causes compilation error

F - Only Line n1 causes compilation error

Working with Inheritance - 37

Consider below codes of 3 java files:

```
//Profitable1.java
package com.training.oca;

public interface Profitable1 {
    default double profit() {
        return 12.5;
    }
}

//Profitable2.java
package com.training.oca;

public interface Profitable2 {
    default double profit() {
        return 25.5;
    }
}

//Profit.java
package com.training.oca;

public abstract class Profit implements Profitable1, Profitable2 {
    /*INSERT*/
}
```

Which of the following needs to be done so that there is no compilation error?

A - Replace /*INSERT*/ with below code:

```
double profit() {
  return 50.0;
}
```

B - No need for any modifications, code compiles as is

C - Replace /*INSERT*/ with below code:

```
public default double profit() {
  return 50.0;
}
```

D - Replace /*INSERT*/ with below code:

```
public double profit() {
  return Profitable2.super.profit();
}
```

E - Replace /*INSERT*/ with below code:

```java
public double profit() {
  return Profitable1.profit();
}
```

F - Replace /*INSERT*/ with below code:

```java
protected double profit() {
  return 50.0;
}
```

Working with Inheritance - 38

Consider below code snippet:

```
interface ILog {
    default void log() {
        System.out.println("ILog");
    }
}

abstract class Log {
    public static void log() {
        System.out.println("Log");
    }
}

class MyLogger extends Log implements ILog {}
```

Which of the following statements is correct?

A - There is no compilation error in the above code

B - There is a compilation error in abstract class Log

C - There is a compilation error in interface ILog

D - There is a compilation error in MyLogger class

Working with Inheritance - 39

Consider below codes of 3 java files:

```
//Super.java
package com.training.oca;

public interface Super {
    String name = "SUPER"; //Line n1
}

//Sub.java
package com.training.oca;

public interface Sub extends Super { //Line n2

}

//Test.java
package com.training.oca;

public class Test {
    public static void main(String[] args) {
        Sub sub = null;
        System.out.println(sub.name); //Line n3
    }
}
```

Which of the following statements is correct?

A - Test class compiles successfully and on execution prints SUPER on to the console

B - Line n3 causes compilation error

C - Line n3 throws an exception at runtime

D - Line n2 causes compilation error

E - Line n1 causes compilation error

Working with Inheritance - 40

Given code of Test.java file:

```java
class Lock {
    public void open() {
        System.out.println("LOCK-OPEN");
    }
}

class Padlock extends Lock {
    public void open() {
        System.out.println("PADLOCK-OPEN");
    }
}

class DigitalPadlock extends Padlock {
    public void open() {
        /*INSERT*/
    }
}

public class Test {
    public static void main(String[] args) {
        Lock lock = new DigitalPadlock();
        lock.open();
    }
}
```

Which of the following options, if used to replace /*INSERT*/, will compile successfully and on execution will print LOCK-OPEN on to the console?

A - None of the other options

B - (Lock)super.open();

C - super.open();

D - ((Lock)super).open();

E - super.super.open();

Working with Inheritance - 41

Consider below code of Test.java file:

```
interface Profitable {
    double profitPercentage = 42.0;
}

class Business implements Profitable {
    double profitPercentage = 50.0; //Line n1
}

public class Test {
    public static void main(String[] args) {
        Profitable obj = new Business(); //Line n2
        System.out.println(obj.profitPercentage); //Line n3
    }
}
```

What will be the result of compiling and executing Test class?

A - Line n1 causes compilation error

B - Line n2 causes compilation error

C - Test class compiles successfully and on execution prints 42.0 on to the console

D - Line n3 causes compilation error

E - Test class compiles successfully and on execution prints 50.0 on to the console

Working with Inheritance - 42

Consider below code snippet:

```
interface Workable {
    void work();
}

/*INSERT*/ {
    public void work() {} //Line n1
}
```

And the statements:

1. abstract class Work implements Workable
2. class Work implements Workable
3. interface Work extends Workable
4. abstract interface Work extends Workable
5. abstract class Work

How many statements can replace /*INSERT*/ such that there is no compilation error?

A - Two statements

B - Three statements

C - Four statements

D - One statement

E - Five statements

Working with Inheritance - 43

Consider below codes of 3 java files:

```java
//Shrinkable.java
package com.training.oca;

public interface Shrinkable {
    public static void shrinkPercentage() {
        System.out.println("80%");
    }
}


//AntMan.java
package com.training.oca;

public class AntMan implements Shrinkable { }

//Test.java
package com.training.oca;

public class Test {
    public static void main(String[] args) {
        AntMan.shrinkPercentage();
    }
}
```

A - There is no compilation error and on execution, Test class prints 80% on to the console

B - There is a compilation error in Shrinkable.java file

C - There is a compilation error in AntMan.java file

D - There is a compilation error in Test.java file

Working with Inheritance - 44

Consider below codes of 3 java files:

```java
//Buyable.java
package com.training.oca;

public interface Buyable {
    int salePercentage = 85;

    public static String salePercentage() {
        return salePercentage + "%";
    }
}

//Book.java
package com.training.oca;

public class Book implements Buyable {}

//Test.java
package com.training.oca;

public class Test {
    public static void main(String[] args) {
        Buyable [] arr = new Buyable[2];
        for(Buyable b : arr) {
            System.out.println(b.salePercentage); //Line n1
            System.out.println(b.salePercentage()); //Line n2
        }

        Book [] books = new Book[2];
        for(Book b : books) {
            System.out.println(b.salePercentage); //Line n3
            System.out.println(b.salePercentage()); //Line n4
        }
    }
}
```

Which of the following statements are correct?

Select ALL that apply

A - There is a compilation error in Book.java file

B - There is a compilation error at Line n2

C - There is a compilation error at Line n1

D - There is a compilation error in Buyable.java file

E - There is a compilation error at Line n3

F - There is a compilation error at Line n4

Working with Inheritance - 45

Given code of Test.java file:

```java
class X {
    void A() {
        System.out.print("A");
    }
}

class Y extends X {
    void A() {
        System.out.print("A-");
    }

    void B() {
        System.out.print("B-");
    }

    void C() {
        System.out.print("C-");
    }
}

public class Test {
    public static void main(String[] args) {
        X obj = new Y(); //Line n1
        obj.A(); //Line n2
        obj.B(); //Line n3
        obj.C(); //Line n4
    }
}
```

What will be the result of compiling and executing above code?

A - Compilation error in class Test

B - Compilation error in class Y

C - A-B-C-

D - AB-C-

Working with Inheritance - 46

Given code of Test.java file:

```java
interface M {
    public static void log() {
        System.out.println("M");
    }
}

abstract class A {
    public static void log() {
        System.out.println("N");
    }
}

class MyClass extends A implements M {}

public class Test {
    public static void main(String[] args) {
        M obj1 = new MyClass();
        obj1.log(); //Line n1

        A obj2 = new MyClass();
        obj2.log(); //Line n2

        MyClass obj3 = new MyClass();
        obj3.log(); //Line n3
    }
}
```

Which of the following statements is correct?

A - Line n3 causes compilation error

B - There is a compilation error in interface M

C - Line n2 causes compilation error

D - Line n1 causes compilation error

E - Given code compiles successfully

F - There is a compilation error in class A

Working with Inheritance - 47

Consider below code of Test.java file:

```java
class Super {
    Super() {
        System.out.print("Reach");
    }
}

class Sub extends Super {
    Sub() {
        Super();
        System.out.print("Out");
    }
}

public class Test {
    public static void main(String[] args) {
        new Sub();
    }
}
```

What will be the result of compiling and executing above code?

A - It prints ReachOut on to the console

B - It prints OutReach on to the console

C - Compilation Error in Super class

D - Compilation Error in Sub class

E - Compilation Error in Test class

Working with Inheritance - 48

Given code of Test.java file:

```java
interface Document {
    default String getType() {
        return "TEXT";
    }
}

interface WordDocument extends Document {
    String getType();
}

class Word implements WordDocument {}

public class Test {
    public static void main(String[] args) {
        Document doc = new Word(); //Line n1
        System.out.println(doc.getType()); //Line n2
    }
}
```

Which of the following statements is correct?

A - Interface WordDocument causes compilation error

B - Interface Document causes compilation error

C - Test class compiles successfully and on execution prints TEXT on to the console

D - Class Word causes compilation error (Correct)

Working with Inheritance - 49

Consider below code of Test.java file:

```
class Currency {
    String notation = "-"; //Line n1

    String getNotation() { //Line n2
        return notation;
    }
}

class USDollar extends Currency {
    String notation = "$"; //Line n3

    String getNotation() { //Line n4
        return notation;
    }
}

class Euro extends Currency {
    protected String notation = "€"; //Line n5

    protected String getNotation() { //Line n6
        return notation;
    }
}

public class Test {
    public static void main(String[] args) {
        Currency c1 = new USDollar();
        System.out.println(c1.notation + ":" + c1.getNotation());

        Currency c2 = new Euro();
        System.out.println(c2.notation + ":" + c2.getNotation());
    }
}
```

What will be the result of compiling and executing above code?

A -

```
-:-
-:-
```

B -

```
$:$
€:€
```

C -

```
-:$
-:€
```

D - Compilation error in Euro class

E - Compilation error in USDollar class

Working with Inheritance - 50

Given code of Test.java file:

```
class Base {
    String msg = "INHALE"; //Line n1
}

class Derived extends Base {
    Object msg = "EXHALE"; //Line n2
}

public class Test {
    public static void main(String[] args) {
        Base obj1 = new Base(); //Line n3
        Base obj2 = new Derived(); //Line n4
        Derived obj3 = (Derived) obj2; //Line n5
        String text = obj1.msg + "-" + obj2.msg + "-" + obj3.msg;
        //Line n6
        System.out.println(text); //Line n7
    }
}
```

What will be the result of compiling and executing above code?

A - None of the other optionsa

B - It executes successfully and prints INHALE-INHALE-EXHALE

C - Line n2 causes compilation error

D - It executes successfully and prints INHALE-EXHALE-EXHALE

E - It executes successfully and prints INHALE-INHALE-INHALE

F - Line n6 causes compilation error

G - Line n5 throws Exception at runtime

Working with Inheritance - 51

Given code of Test.java file:

```
interface Rideable {
    void ride(String name);
}

class Animal {}

class Horse extends Animal implements Rideable {
    public void ride(String name) {
        System.out.println(name.toUpperCase() + " IS RIDING THE
         HORSE");
    }
}

public class Test {
    public static void main(String[] args) {
        Animal horse = new Horse();
        /*INSERT*/
    }
}
```

Which of the following options, if used to replace /*INSERT*/, will compile successfully and on execution will print EMMA IS RIDING THE HORSE on to the console?

Select ALL that apply.

A - (Horse)(Rideable)horse.ride(“EMMA’”);

B - ((Rideable)horse).ride(““emma’);

C - horse.ride(“EMMA”);

D - ((Horse)(Rideable)horse).ride(“emma’);

E - (Rideable)(Horse)horse.ride(““EMMA”);

F - ((Rideable)(Horse)horse).ride(“EMMA”);

G - ((Horse)horse).ride(“Emma’”);

H - (Horse)horse.ride(“EMMA”);

Working with Inheritance - 52

Given code of Test.java file:

```java
class Parent {
    String quote = "MONEY DOESN'T GROW ON TREES";
}

class Child extends Parent {
    String quote = "LIVE LIFE KING SIZE";
}

class GrandChild extends Child {
    String quote = "PLAY PLAY PLAY";
}

public class Test {
    public static void main(String[] args) {
        GrandChild gc = new GrandChild();
        System.out.println(/*INSERT*/);
    }
}
```

Which of the following options, if used to replace /*INSERT*/, will compile successfully and on execution will print MONEY DOESN'T GROW ON TREES on to the console?

Select ALL that apply.

A - (Parent)(Child)gc.quote

B - gc.quote

C - (Parent)gc.quote

D - ((Parent)(Child)gc).quote

E - ((Parent)gc).quote

Working with Inheritance - 53

Consider below codes of 2 java files:

```
//Flyable.java
package com.training.oca;

public interface Flyable {
    static int horizontalDegree() { //Line n1
        return 20;
    }

    default void fly() {
        System.out.println("Flying at " + horizontalDegree() + "
         degrees."); //Line n2
    }

    void land();
}

//Aeroplane.java
package com.training.oca;

public class Aeroplane implements Flyable {
    public void land() {
        System.out.println("Landing at " + -
         Flyable.horizontalDegree() + " degrees."); //Line n3
    }

    public static void main(String[] args) {
        new Aeroplane().fly();
        new Aeroplane().land();
    }
}
```

What will be the result of compiling and executing Aeroplane class?

A - Compilation error at Line n2

B - Compilation error at Line n1

C - Given code compiles successfully and on execution prints below in the output:

```
Flying at 20 degrees.
Landing at -20 degrees.
```

D - Compilation error at Line n3

Working with Inheritance - 54

Consider below codes of 2 java files:

```java
//GetSetGo.java
package com.training.oca;

public interface GetSetGo {
    int count = 1; //Line n1
}

//Test.java
package com.training.oca;

public class Test {
    public static void main(String[] args) {
        GetSetGo [] arr = new GetSetGo[5]; //Line n2
        for(GetSetGo obj : arr) {
            obj.count++; //Line n3
        }
        System.out.println(GetSetGo.count); //Line n4
    }
}
```

Which of the following statements is correct?

A - Line n2 causes compilation error

B - Test class compiles successfully and on execution prints 6 on to the console

C - Test class compiles successfully and on execution prints 5 on to the console

D - Line n1 causes compilation error

E - Line n3 causes compilation error

F - Line n4 causes compilation error

Working with Inheritance - 55

Consider below code of Test.java file:

```
class Super {
    public String num = "10"; //Line n1
}

class Sub extends Super {
    protected int num = 20; //Line n2
}

public class Test {
    public static void main(String[] args) {
        Super obj = new Sub();
        System.out.println(obj.num += 2); //Line n3
    }
}
```

What will be the result of compiling and executing above code?

A - It executes successfully and prints 202 on to the console

B - Compilation error at Line n3

C - Compilation error at Line n2

D - It executes successfully and prints 12 on to the console

E - It executes successfully and prints 102 on to the console

F - It executes successfully and prints 22 on to the console

Working with Inheritance - 56

Consider below code of Test.java file:

```java
class MyClass {
    MyClass() {
        System.out.println(101);
    }
}

class MySubClass extends MyClass {
    final MySubClass() {
        System.out.println(202);
    }
}
public class Test {
    public static void main(String[] args) {
        System.out.println(new MySubClass());
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - 202 101 <Some text containing @ symbol>

C - 101 202 <Some text containing @ symbol>

D - 202 <Some text containing @ symbol>

E - 101 <Some text containing @ symbol>

Working with Inheritance - 57

Consider below code of Test.java file:

```java
interface Perishable1 {
    default int maxDays() {
        return 1;
    }
}

interface Perishable2 extends Perishable1 {
    default int maxDays() {
        return 2;
    }
}

class Milk implements Perishable2, Perishable1 {}

public class Test {
    public static void main(String[] args) {
        Perishable1 obj = new Milk();
        System.out.println(obj.maxDays());
    }
}
```

Which of the following statements is correct?

A - Given code compiles successfully and on execution Test class prints 2 on to the console

B - Interface Perishable2 causes compilation error

C - Class Milk causes compilation error

D - Class Test causes compilation error

E - Given code compiles successfully and on execution Test class prints 1 on to the console

Working with Inheritance - 58

Consider below codes of 4 java files:

```
//I1.java
package com.training.oca;

public interface I1 {
    int i = 10;
}

//I2.java
package com.training.oca;

public interface I2 {
    int i = 20;
}

//I3.java
package com.training.oca;

public interface I3 extends I1, I2 { //Line n1

}

//Test.java
package com.training.oca;

public class Test {
    public static void main(String[] args) {
        System.out.println(I1.i); //Line n2
        System.out.println(I2.i); //Line n3
        System.out.println(I3.i); //Line n4
    }
}
```

Which of the following statements is correct?

A - Line n3 causes compilation error

B - Line n2 causes compilation error

C - Line n1 causes compilation error

D - Line n4 causes compilation error

E - There is no compilation error

Working with Inheritance - 59

Given code of Test.java file:

```
class Paper {
    static String getType() { //Line n1
        return "GENERIC";
    }
}

class RuledPaper extends Paper {
    String getType() { //Line n2
        return "RULED";
    }
}

public class Test {
    public static void main(String[] args) {
        Paper paper = new RuledPaper(); //Line n3
        System.out.println(paper.getType()); //Line n4
    }
}
```

Which of the following statements is true for above code?

A - Compilation error in Test class

B - Code compiles successfully and on execution prints RULED on to the console

C - Compilation error in RuledPaper class

D - Code compiles successfully and on execution prints GENERIC on to the console

Working with Inheritance - 60

Given code of Test.java file:

```
class Base {
    int id = 1000; //Line n1

    Base() {
        Base(); //Line n2
    }

    void Base() { //Line n3
        System.out.println(++id); //Line n4
    }
}

class Derived extends Base {
    int id = 2000; //Line n5

    Derived() {} //Line n6

    void Base() { //Line n7
        System.out.println(--id); //Line n8
    }
}

public class Test {
    public static void main(String[] args) {
        Base base = new Derived(); //Line n9
    }
}
```

What will be the result of compiling and executing above code?

A - 0

B - 2001

C - 1001

D - 999

E - Compilation error

F - 2000

G - An exception is thrown

H - -1

Working with Inheritance - 61

Consider below code of Test.java file:

```java
interface I1 {
    public static void print(String str) {
        System.out.println("I1:" + str.toUpperCase());
    }
}

class C1 implements I1 {
    void print(String str) {
        System.out.println("C1:" + str.toLowerCase());
    }
}

public class Test {
    public static void main(String[] args) {
        I1 obj = new C1();
        obj.print("Java");
    }
}
```

Which of the following statements is correct?

A - Given code compiles successfully and on execution prints 11:JAVA on to the console

B - Class C1 causes compilation error

C - Class Test causes compilation error

D - Interface 11 causes compilation error

E - Given code compiles successfully and on execution prints C1:java on to the console

Working with Inheritance - 62

Given code of Test.java file:

```java
class Super {
    final int NUM = -1; //Line n1
}

class Sub extends Super {
    /\*INSERT\*/
}

public class Test {
    public static void main(String[] args) {
        Sub obj = new Sub();
        obj.NUM = 200; //Line n2
        System.out.println(obj.NUM); //Line n3
    }
}
```

Above code causes compilation error, which modifications, done independently, enable the code to compile and on execution print 200 on to the console?

Select ALL that apply.

A - Replace /*INSERT*/ with boolean NUM;

B - Replace /*INSERT*/ with float NUM;

C - Replace /*INSERT*/ with Object NUM;

D - Replace /*INSERT*/ with short NUM;

E - Replace /*INSERT*/ with byte NUM;

F - Replace /*INSERT*/ with double NUM;

G - Remove final modifier from Line n1

H - Replace /*INSERT*/ with int NUM;

Working with Inheritance - 63

Consider below code of Test.java file:

```java
abstract class Log {
    abstract long count(); //Line n1
    abstract Object get(); //Line n2
}

class CommunicationLog extends Log {
    int count() { //Line n3
        return 100;
    }

    String get() { //Line n4
        return "COM-LOG";
    }
}

public class Test {
    public static void main(String[] args) {
        Log log = new CommunicationLog(); //Line n5
        System.out.print(log.count());
        System.out.print(log.get());
    }
}
```

Which of the following statement is correct?

A - Line n5 causes compilation error

B - Line n3 causes compilation error

C - Line n4 causes compilation error

D - Given code compiles successfully and on execution prints LOOCOM-LOG on to the console